Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (currently amended) A device comprising:

a memory unit including executable software;

a plurality of class files stored in the memory unit; and,

a computing unit connected to the memory unit, the computing unit being able to execute a Java Virtual Machine, the computing unit configured to execute software for generating two or more files from the plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space,

wherein the generated files are directly interpretable by the Java Virtual Machine, and wherein the number of the generated files is less than the number of the plurality of class files, and wherein a given generated file comprises: a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries; a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files; and, a fixup table for providing information to the Java Virtual Machine for resolving at least one entry in the given generated file at link time, and at least two of the generated files are generated as sibling files in a common sibling group, wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group, wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets, and wherein references to files that are not part of the common sibling group are indicated using symbolic references.

2. (previously presented) The device of claim 1, wherein the information in the fixup table comprises the location of data needed for resolving a symbolic reference in the given generated file.

3. (cancelled).

4. (previously presented) The device of claim 1, wherein for the given generated file, the byte codes and information structure comprises a second hard offset for cross-referencing a method included in the given generated file that was previously symbolically referenced.

5. (previously presented) The device of claim 1, wherein at least one of the hard offsets does not need to be resolved or put into context by the Java Virtual Machine at link time.

6. (cancelled).

7. (currently amended) A method for generating two or more files from a plurality of class files having constant pools on a computing unit by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the generated files are directly interpretable by a Java Virtual Machine, and wherein the number of the generated files is less than the number of the plurality of class files, and for a given generated file the method comprises:

identifying class files with common entries in the constant pools of the class files;

generating a constant pool for the given generated file by combining constant pool entries from the plurality of class files with common entries without duplication;

generating the byte codes and information structure for the given generated file by combining byte codes and information structure entries from the plurality of class files; and,

generating a fixup table for providing information to [[a]] the Java Virtual Machine for resolving at least one entry in the given generated file at link time; and

wherein the method further comprises generating at least two of the generated files as sibling files in a common sibling group by providing a sibling list for each of the sibling files listing other sibling files in the common sibling group, wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets, and wherein references to files that are not part of the common sibling group are indicated using symbolic references.

8. (previously presented) The method of claim 7, wherein the method further comprises providing location of data needed for resolving a symbolic reference in the given generated file as the information in the fixup table.

9. (cancelled).

10. (previously presented) The method of claim 7, wherein the method further comprises, for the given generated file, providing a second hard offset in the byte codes and information structure for cross-referencing a method now included in the given generated file that was previously symbolically referenced.

11. (previously presented) The method of claim 7, wherein the method further comprises providing at least one of the hard offsets in a manner that does not need to be resolved or put into context by the Java Virtual Machine at link time.

12. (cancelled).

13. (currently amended) A storage medium storing computer executable instructions that when executed by a computing unit generates one or more files from a plurality of class files by combining elements from the plurality of class files without duplication of entries for reducing storage space, wherein the generated files are directly interpretable

by a Java Virtual Machine, and wherein the executable software comprises code for generating a given generated file to include:

a constant pool created by combining constant pool entries from two or more of the plurality of class files without duplication of entries;

a byte codes and information structure created by combining byte codes and information structure entries from the two or more of the plurality of class files; and,

a fixup table for providing information to [[a]] the Java Virtual Machine for resolving at least one component of the given generated file at link time;

wherein the executable software further comprises code for generating at least two of the generated files as sibling files in a common sibling group, wherein each of the sibling files comprise a sibling list for listing other sibling files in the common sibling group, wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets, and wherein references to files that are not part of the common sibling group are indicated using symbolic references.

14. (previously presented) The storage medium of claim 13, wherein the information in the fixup table comprises the location of data needed for resolving a symbolic reference in the given generated file.

15. (cancelled).

16. (previously presented) The storage medium of claim 13, wherein for the given generated file, the executable software comprises code for generating a second hard offset in the byte codes and information structure for cross-referencing a method included in the given generated file that was previously symbolically referenced.

17. (previously presented) The storage medium of claim 13, wherein for the given generated file, at least one of the hard offsets does not need to be resolved or put into context by the Java Virtual Machine at link time.

18. (cancelled).

19. (previously presented) The device of claim 1, wherein the information in the fixup table of a given sibling file comprises the location of data of a cross-referenced sibling file to place one of the hard offsets that corresponds to the cross-referenced sibling file into context at link time.

20. (previously presented) The method of claim 7, wherein the method further comprises providing the location of data of a cross-referenced sibling file as the information in the fixup table of a given sibling file to place one of the hard offsets that corresponds to the cross-reference sibling file into context at link time.

21. (previously presented) The storage medium of claim 13, wherein the information in the fixup table of a given sibling file comprises the location of data of a cross-referenced sibling file to place one of the hard offsets that corresponds to the cross-referenced sibling file into context at link time.

22. (previously presented) The device of claim 1, wherein as few sibling files are generated as possible while satisfying a constraint on individual generated file size.

23. (previously presented) The method of claim 7, wherein the method further comprises generating as few sibling files as possible while satisfying a constraint on individual generated file size.

24. (previously presented) The storage medium of claim 13, wherein as few sibling files are generated as possible while satisfying a constraint on individual generated file size.